Software Design Document

<Project Name>

s5273814 – Nathanael Gazzard

s5309988 - Stephen Urquhart

Table of Contents

[1.0 System Vision 3](#_Toc46748622)

[1.1 Problem Background 3](#_Toc46748623)

[1.2 System Overview 3](#_Toc46748624)

[1.3 Potential Benefits 3](#_Toc46748625)

[2.0 Requirements 4](#_Toc46748626)

[2.1 User Requirements 4](#_Toc46748627)

[2.2 Software Requirements 4](#_Toc46748628)

[2.3 Use Cases 4](#_Toc46748629)

[3.0 System Components and Software Design 5](#_Toc46748630)

[3.1 System Components 5](#_Toc46748631)

[3.2 Software Design 5](#_Toc46748632)

[4.0 User Interface Design 6](#_Toc46748633)

# System Vision

## Problem Background

Kaggle.com hosts a dataset of ~400k restaurant inspections of New York City (NYC) food establishments. The inspections span from January 2010 to August 2017 and are graded on an A-F scale, having been collected by the NYC Department of Health. A Data Analysis and Visualization Tool (DAVT) must be developed that allow user to make the following five queries:

1. Retrieve all inspection details for a specified period.
2. Plot violations distributed on a per-suburb basis.
3. Retrieve all violations containing a specified keyword for a specified period.
4. Map violations pertaining to animals distributed over time and suburbs.
5. Pull the 100 places with the best improvement over the last year for each boro.

## System Overview

The system will have three distinct views:

1. The home page from which the type of query is selected, and refining details specified.
2. The data result page which lists the results of the selected query.
3. The data visualization page which visualises the results of the data result page.

The home page can be returned to from either of the other pages to commence a new query instantly. The system will use clean minimal graphics keeping the functionality foremost.

## Potential Benefits

The system will make accessing the desired data a straightforward process, with no distractions along the way.

It will allow for quickly switching back and forth between a raw data view and data visualisation to make comparisons.

It will make analysing the immense amount of data accessible without overwhelming the user.

# Requirements

## User Requirements

In this section you detail how a user is supposed to interact with or use your program. What do they ***need*** to be able to do? This should all be from the end users perspective. Can be a combination of narrative text and listing of needs.

**Assignment note: You have not been given a client/user, so you can make one up. Who do you think would be using your software?**

**Overview:** This Data Analysis and Visualization Tool is designed to empower restaurant owners and health inspectors in New York with a user-friendly solution to analyse and visualise restaurant inspection results. As a health inspector, Mary Smith, you need a seamless understanding of inspection data trends and make informed decisions to ensure public health safety.

**User Persona:** Mary Smith is a dedicated health inspector responsible for conducting restaurant inspections across New York neighbourhoods. Her role requires her to analyse inspection data, identify trends, and take necessary actions to maintain food safety standards.

**Interaction and Usage:**

1. *Intuitive Dashboard:* Upon opening the software, Mary encounters an intuitive dashboard showcasing key metrics and visualisations of inspection data. This dashboard provides a platform that is easy to navigate and supports a linear workflow.
2. *Flexible Period Selection:* Mary can easily select a specific period for analysis. The tool allows her to choose a date range, enabling her to focus on recent or historical data.
3. *Violation Distribution Plot:* Mary navigates to the "Violation Distribution" section. Here, she selects a date range and instantly views a graphical representation of violation distribution across different suburbs. This helps her identify areas with higher instances of violations.
4. *Keyword-Based Violation Search:* In the "Keyword Search" feature, Mary enters keywords like "rodent" or "contamination." The tool displays a list of violations matching her search, aiding her in targeted investigation.
5. *Animal-Related Analysis:* Mary accesses the "Animal Analysis" section to analyse cases related to rodents, pests, and other animals. She can view trends over time and explore how animal-related violations are distributed across neighbourhoods.

**User Needs:**

* Easily select specific periods for analysis.
* Visualize violation distribution over neighbourhoods.
* Search for violations based on keywords.
* Analyse trends related to animal-related violations.

The Data Analysis and Visualization Tool caters to the needs of health inspectors like Mary Smith, enabling them to make data-driven decisions to enhance food safety standards across New York's restaurants.

## Software Requirements

In this section you detail what the requirements for the software are. What functionality will it provide? This is usually a formal listing, with requirements often using the word ‘Shall’. IE:

R1.1 The program shall accept multiple file names as arguments from the command line.

R1.2 Each file name can be a simple file name or include the full path of the file with one or more levels. etc …

Can be primarily functional requirements, though you may include other types if you think of them.

**R1.1** The tool shall provide a user-friendly graphical user interface (GUI) for easy interaction and navigation.

**R1.2** The GUI shall include a dashboard displaying key metrics and visualisations of inspection data.

**R1.3** Users shall be able to select a specific period for analysis using date range selectors.

**R2.1** The tool shall offer a "Violation Distribution" feature that generates a graphical plot showcasing the distribution of violations across different suburbs for the selected period.

**R2.2** A "Keyword Search" functionality shall allow users to enter keywords and retrieve a list of violations containing those keywords for the selected period.

**R2.3** An "Animal Analysis" section shall enable users to analyse trends related to animal-related violations (e.g., rodents, pests) over time and their distribution across neighbourhoods.

**R3.1** The system shall utilise a version control system (VCS) such as GitHub to track changes in the source code and collaborate on development.

**R4.1** The tool's GUI shall have a responsive design, ensuring usability on different screen sizes and devices.

**R4.2** User preferences, such as selected date ranges and visualisation settings, shall be saved and restored upon subsequent logins.

**R5.1** The tool shall implement appropriate security measures to safeguard user data and prevent unauthorised access.

**R5.2** The backend APIs shall efficiently handle data retrieval, processing, and storage for various analysis tasks.

**R5.3** The database shall store inspection data, user preferences, and other relevant information securely and efficiently.

**R5.4** The tool shall support concurrent usage by multiple users without compromising performance.

**R5.5** The backend shall be scalable to accommodate growing user numbers and increasing data volumes.

**R6.1** The system shall maintain a detailed audit trail of user interactions and system activities for accountability and troubleshooting.

**R6.2** The tool shall ensure the accuracy and consistency of data used for analysis and visualisation tasks.

**R6.3** The GUI shall provide clear and intuitive labels, tooltips, and error messages to guide users during interaction.

## Use Cases & Use Case Diagrams

In this section, you provide some use cases showing how people may use your software.

**Use Case 1: Restaurant Owner - Trend Analysis**

User: Sarah, a restaurant owner.

Scenario: Sarah wants to track the inspection trends for her restaurant over the past year.

Steps:

1. Sarah logs into the tool.
2. She selects the date range from the past year.
3. Sarah clicks on the "Violation Distribution" feature.
4. The tool generates a plot showing the distribution of violations in different suburbs for the selected period.
5. Sarah notices increased violations in a specific suburb and takes corrective actions.

**Use Case 2: Health Inspector – Keyword Search**

User: John, a health inspector.

Scenario: John needs to investigate cases related to “rodents” reported last month.

Steps:

1. John accesses the tool’s “Keyword Search” feature.
2. He enters the keyword “rodents” and selects the last month as the date range.
3. The tool retrieves and displays a list of violations containing the keyword “rodents.”
4. John identifies patterns in the violations and decides to conduct targeted inspections in the affected areas.

**Use Case 3: Analyst - Animal-Related Analysis**

User: Lisa, a data analyst.

Scenario: Lisa is tasked with analysing the trend of animal-related violations across neighbourhoods.

Steps:

1. Lisa logs into the tool.
2. She goes to the "Animal Analysis" section.
3. Lisa selects the past two years as the date range.
4. The tool presents visualisations depicting the trend of animal-related violations over time.
5. Lisa identifies neighbourhoods with consistent animal-related issues and suggests targeted awareness campaigns.

# Software Design and System Components

## Software Design

A block diagram/flowchart of how your software might work

## System Components

### Functions

Preliminary list of all functions in the software. For each function in the list the following information is provided:

* a brief description of what it does (1 or 2 sentences);
* a list of the input parameters, and their data types, and what they are used for;
* a list of any side effects caused by the function (ie change global or member variables, changes data passed by reference from calling function etc)
* a description of the function’s return value

### Data Structures / Data Sources

List of all data structures in the software (eg linked lists, trees, arrays etc) or eternal data sources. For each data structure in the list the following information is provided:

* Type of structure (tree, list etc),
* Description of where and how it is used
* List of data members, and what each one is for do
* List of functions that use it

### Detailed Design

Pseudocode for all non-standard / non-trivial algorithms that operate on data structures

# User Interface Design

Group discussion led to the consensus that this system would be best served by an aesthetically clean and minimal design that would make the interface and data easy to parse. Initial sketches were created on paper to figure out the placement of each item before the final wireframes were created in draw.io.

## Visual Design

### Elements:

The title of the system will be displayed as a consistent header on all pages. \*

The “current query >” button will only appear if the user has previously made a query. When pressed, it will open the data page with the last query that was run.

The “query” field has a drop-down menu from which the user can select one of the five predefined queries. Upon doing so, the form beneath will update to allow the user to enter the contextually appropriate query parameters.

The "prior search details" link in the query details form will open a floating list of date-time stamps representing the user's prior query parameters for the current query type. The user can select one that will fill the form with the former values.

Pressing the submit button will run the query and take the user to the data results page provided the required parameters have been supplied.

The data and visualisation view can be toggled like browser tabs.

When a query or visualisation takes up more room than is available on screen, the display area will be scrollable.

The new query button on the data or visualisation page will take the user back to the home (query) page. The query field and parameters form will be prefilled with the details of the query they were just viewing.

\*Titles and names are subject to change and should be considered placeholders.

### Wireframe:

A group of blueprints

Description automatically generated

### Navigation (PC):

A screenshot of a computer screen

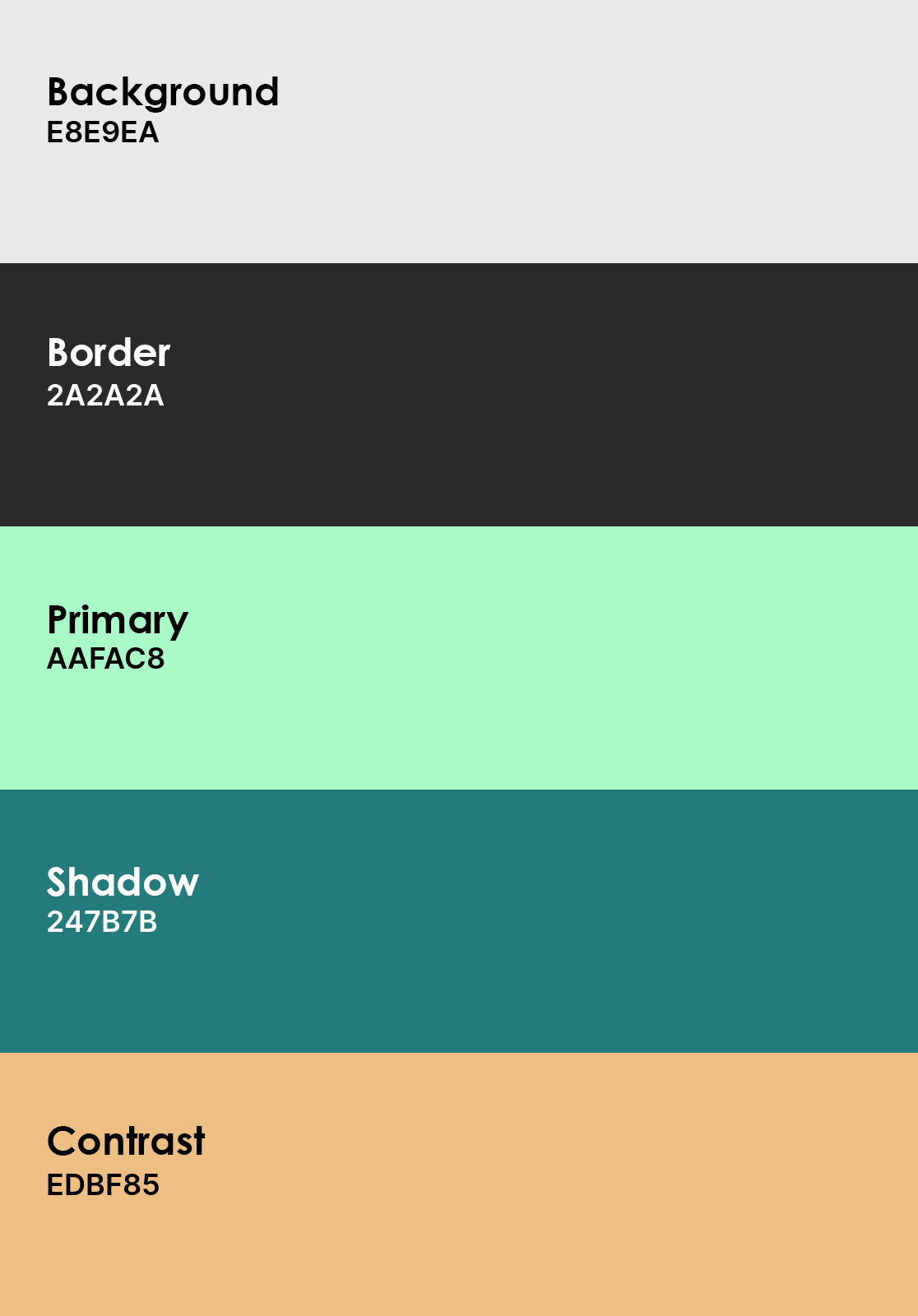
Description automatically generated

### Navigation (Mobile):

A screenshot of a computer

Description automatically generated

### Colour Palette:



### Graphic Sample:

A green rectangular sign with black text

Description automatically generated

### Font:

In line with the goal of clean design in mind, research was done into what font is the easiest to parse in a digital format. Sans-serif fonts were ubiquitous in their domination of this space, but foremost among them was Helvetica. However, a recurrent trend in the reading was Roboto as a suitable alternative. While Roboto is technically less minimalist than Helvetica, its liberties are in service of readability. Additionally, with the prospect of this system being utilised in mobile spaces, particularly Android devices, Roboto becomes a direct continuation of the native device experience. For these reasons, Roboto was selected over Helvetica.

Sample of the selected font, Roboto.